revibio.app

Fejlesztői dokumentáció

##### Alapkövetelmények

A webalkalmazás Node.js-el (v22.0.0), frontend Vue (Vue 3) keretrendszerrel készült. A fejlesztés során TypeScript-et használtunk.

A backend fejlesztéséhez Nest.js-t használtunk az adatbázis pedig MongoDB, az adatbázis futtatására Dockert használtunk.

Frontend banckend kommunikáció

Futtatáshoz szükséges szoftverek:

* Node 16.0.0 vagy annál újabb
* npm 8 vagy annál újabb
* Docker
* webböngésző
  + Chrome 87+
  + Edge 88+
  + Firefox 78+
  + Safari 13+

Frontend futtatáshoz szükséges node modulok:

* tailwindcss: 3.4.17
* axios: 1.7.9
* echarts: 5.6.0
* marked: 15.0.8
* motion: 12.3.1
* pinia: 2.2.4
* tailwindcss-animated: 1.1.2
* uuid: 11.1.0
* vue: 3.5.12
* vue-router: 4.4.5
* motionone/dom: 10.18.0
* motionone/vue: 10.16.4

Backend futtatáshoz szükséges node modulok:

* nestjs/common: 10.0.0
* nestjs/config: 3.3.0
* nestjs/core: 10.0.0
* nestjs/jwt: 10.2.0
* nestjs/mapped-types: \*
* nestjs/mongoose: 10.1.0
* nestjs/platform-express: 10.0.0
* types/jsonwebtoken: 9.0.7
* argon2: 0.41.1
* class-transformer: 0.5.1
* class-validator: 0.14.1
* dotenv: 16.4.5
* jsonwebtoken: 9.0.2
* mongodb: 6.15.0
* mongoose: 8.13.1
* nodemailer: 6.10.0
* reflect-metadata: 0.2.0
* rxjs: 7.8.1
* uuid: 11.1.0

##### Futtatás

A frontend futtatásához le kell klónozni ezt a https://github.com/revi-bio/webapp github repositoryt, ezután a mappába le kell tölteni a node modulokat a pnpm i paranccsal. Ezután az pnpm run dev paranccsal tudjuk elindítani a fejlesztői környezetet.

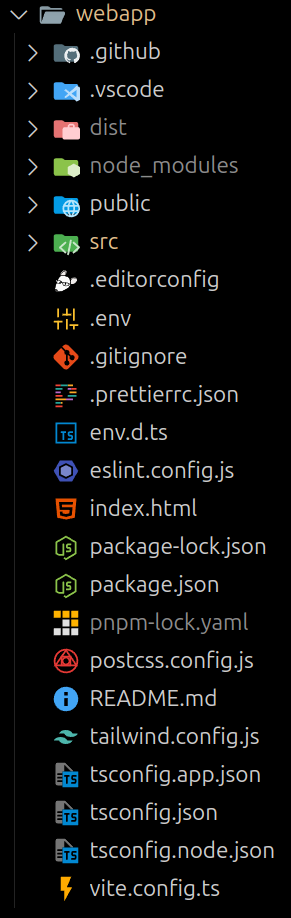
A backend futtatásához le kell klónoznunk ezt a https://github.com/revi-bio/api github repositoryt, ezután a mappába le kell tölteni a node modulokat a pnpm i paranccsal. Ezután az pnpm start:dev paranccsal tudjuk elindítani a fejlesztői környezetet.

A futtatáshoz szükséges, hogy fusson egy mongo docker image ami az adatbázist szolgáltatja.

A backend a http://localhost:3000/, a frontend pedig a http://localhost:5173/ címekel érhetőek el, miután futtatjuk őket.

##### Mappastruktúra

Frontend

A frontend mappájának megnyitásakor a következő mappákat láthatjuk.

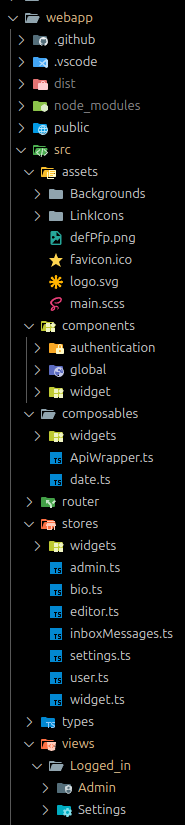
Mappák:

* .github: GitHub-specifikus fájlok.
* .vscode: Visual Studio Code specifikus beállítások.
* dist: Az alkalmazás buildelt (lefordított) verziója ide kerül.
* node\_modules: Az összes node csomag ide kerül telepítéskor.
* public: Statikus fájlok helye, amit a szerver változtatás nélkül szolgál ki.
* src: A forráskód mappája.

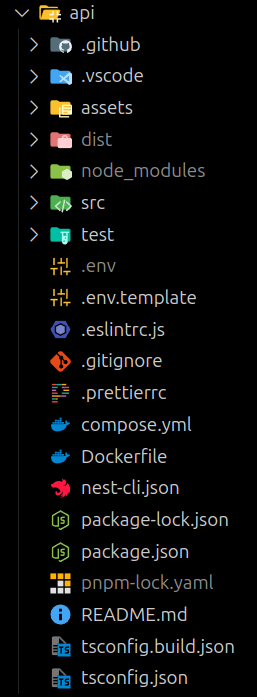
Fájlok:

* .editconfig: Kódformázási beállítások különböző szerkesztők számára.
* .env: Környezeti változók.
* .gitignore: Fájlok és mappák kizárása a verziókezelésbő.
* .prettierc.json: Prettier kódformázó eszköz beállításai.
* env.d.ts: TypeScript deklarációs fájl környezeti változók típusdefinícióihoz.
* eslint.config.js: ESLint kódelemző beállításai.
* index.html: Az alkalmazás alap HTML fájlja, amelybe a Vite/JavaScript betöltődik.
* package-lock.json: Az aktuális csomagverziók lock-fájlja.
* package.json: A projekt metaadatai.
* pnpm-lock.yaml: A pnpm csomagkezelő lock-fájlja.
* postcss.config.js: PostCSS konfiguráció.
* README.md: A projekt leírása, telepítési és futtatási instrukciók.
* tailwind.config.js: Tailwind CSS keretrendszer beállításai.
* tsconfig.app.json: TypeScript beállítások csak az alkalmazás részhez.
* tsconfig.json: Általános TypeScript konfiguráció.
* tsconfig.node.json: TypeScript beállítások Node.js-specifikus részekhez.
* vite.config.ts: Vite build rendszer és dev szerver konfigurációja.

webapp src mappa tartalma:

* assets/ Statikus erőforrások (képek, ikonok, stílusok)
  + Backgrounds/ Háttérképek tárolása.
  + LinkIcons/ Linkekhez tartozó ikonok.
  + defPfP.png Alapértelmezett profilkép.
  + favicon.ico Alkalmazás böngészőfülön megjelenő ikonja.
  + logo.svg Alkalmazás logója SVG formátumban.
  + main.scss Alkalmazás fő SCSS fájlja, ahol globális stílusok és importok vannak.
* components/ Újrafelhasználható Vue komponensek.
  + authentication/ Bejelentkezéssel, regisztrációval, jogosultsággal kapcsolatos komponensek.
  + global/ Globálisan használt UI elemek.
  + widget/ Widgetekhez kapcsolódó komponensek.
* composables/ Vue 3 Composition API egyedi logikai moduljai.
  + widgets/ Widget-specifikus logikai egységek.
  + ApiWrapper.ts API hívásokat kezelő wrapper függvények.
  + date.ts Dátumkezeléssel kapcsolatos segédfüggvények.
* router/ Útvonalkezeléshez szükséges fájlok index.ts.
* stores/ Állapotkezelő modulok.
  + widgets/ Widgetek állapotkezelő moduljai.
  + admin.ts Admin felülethez tartozó store.
  + bio.ts Felhasználó leírásához tartozó állapotkezelés.
  + editor.ts Widget vagy szövegszerkesztő állapot.
  + inboxMessages.ts Bejövő üzenetek store.
  + settings.ts Alkalmazás beállításainak kezelése.
  + user.ts Felhasználói adatok, jogosultságok stb.
  + widget.ts Widgetek közös állapotkezelése.
* types/ TypeScript típusdefiníciók helye. (Pl. User, Message, WidgetConfig interface-ek, type aliasok.)
* views/ Oldalszintű komponensek (page-ek).
  + Logged\_in/ Bejelentkezett felhasználók számára elérhető nézetek.
  + Admin/ Admin felület oldalkomponensei.
  + Settings/ Beállítások oldalkomponensei.

Backend

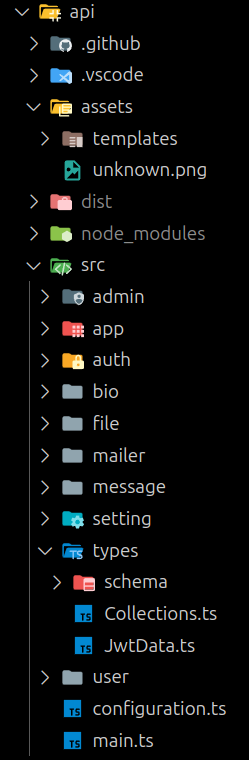
 Mappák:

* .github: GitHub-specifikus fájlok
* .vscode: Visual Studio Code specifikus beállítások
* assets: Statikus erőforrások
* dist: Az alkalmazás buildelt (lefordított) verziója ide kerül
* node\_modules: Az összes NPM csomag ide kerül telepítéskor
* src: A forráskód mappája

Fájol:

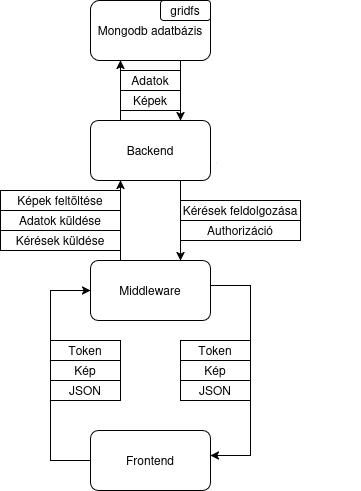
* .env: Futtatási környezeti változók.
* .env.template: Sablon .env fájlhoz mit kell kitölteni új környezet telepítésekor.
* .eslintrc.js: ESLint kódelemző beállításai.
* .gitignore: Fájlok és mappák kizárása a verziókezelésbő.
* .prettierrc: Prettier kódformázó eszköz beállításai.
* compose.yml: Docker Compose fájl több konténeres alkalmazás definiálásához.
* Dockerfile: Leírás, hogyan épüljön fel a Docker konténer az API-hoz.
* nest-cli.json: NestJS CLI konfiguráció, amely meghatározza a build viselkedését, mappaelrendezést.
* package.json: Projekt metaadatai és függőségei.
* package-lock.json / pnpm-lock.yaml: Telepített csomagok pontos verzióinak nyomon követésére szolgál.
* README.md: A projekt dokumentációs fájlja.
* tsconfig.json: TypeScript fordító konfiguráció.
* tsconfig.build.json: TypeScript beállítások csak a build folyamathoz.

api src mappa tartalma:

* admin/ Az admin felhasználókhoz kapcsolódó logikák.
* app/ A fő alkalmazásmodul, ami a projekt belépési pontját foglalja magába – itt történik a többi modul importálása és regisztrálása.
* auth/ Hitelesítéssel kapcsolatos logika.
* bio/ "biográfiai" adatokat kezelő modul.
* file/ Fájlkezeléssel kapcsolatos műveletek.
* mailer/ Emailküldéssel kapcsolatos funkciók.
* message/ Üzenetkezelési modul rendszerüzenetek, értesítések.
* setting/ Rendszer és felhasználói beállítások kezelése.
* types/ Globálisan használt típusdefiníciók.
  + schema/: Struktúrákat és sémákat tartalmaz a táblákhoz.
  + Collections.ts: Adatbázis kollekciók nevét tartalmazza.
  + JwtData.ts: JWT token tartalmának típusdefiníciója.
* user/ Felhasználói logikát tartalmaz:
* configuration.ts: Alkalmazás konfigurációját kezeli.
* main.ts: A NestJS alkalmazás belépési pontja. Itt történik az app elindítása.

##### Az alkalmazás

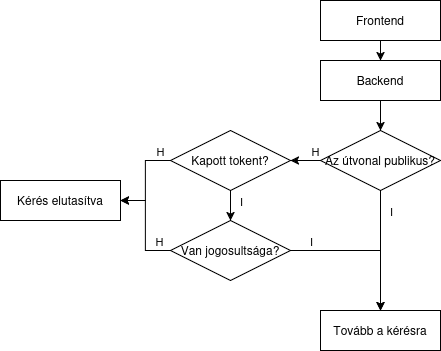
Kommunikáció

Az applikáció három részből áll: frontend, backend, adatbázis.

A frontend a backendnek 3 féle adatot küldhet és kaphat:

* JSON adatszerkezet
* Kép
* Authorizációs tokeneket

A kommunikáció HTTP requestekkel van lebonyolítva. A backend middleware-jei kezelik az authorizationt, kép feltöltéseket és a további kéréseket.

A JWT alapú autentikáció során a felhasználó bejelentkezéskor egy digitálisan aláírt tokent kap, amelyet minden további kérésnél elküld az API-nak. Ez a token tartalmazza a felhasználó azonosító adatait, és a szerver ellenőrzi annak érvényességét minden kérésnél. Ez a módszer lehetővé teszi a stateless autentikációt, mivel a szerver nem tárolja a felhasználói munkamenet állapotát.

Adatbázis

Az adatbázis egy MongoDB adatbázis. GridFs-t használunk.

Backend

A backend a NestJS keretrendszerre épül, amely a Node.js és TypeScript alapú. Az alkalmazás modulokra van bontva, amelyek mindegyike egy adott funkcionális egységet képvisel. Minden modul tartalmazza a saját vezérlőit (controllers), szolgáltatásait (services) és adatátviteli objektumait (DTO-k), amelyek együttműködve biztosítják a modul önállóságát és tesztelhetőségét.

Jelszó titkosításra argon2-t használunk ezzel biztosítva a jelszavak biztonságos tárolását.

##### Frontend

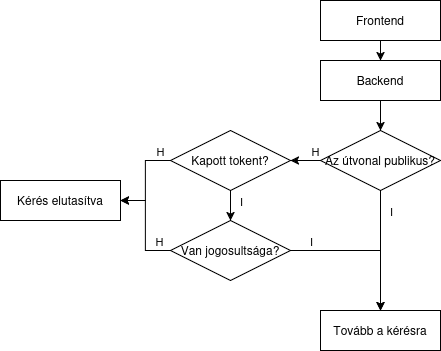
A frontend Vue 3 keretrendszert használ a megjelenésben pedig Tailwind css-t. Ez kommunikál a backenddel.

##### Authorization

Vannak olyan útvonalak amik előzetes authentication nélkül nem adnak választ. Olyan útvonalak is vannak amikhez csak megfelelő jogosultságokkal lehet hozzáférni.

Nem védett útvonalak: login, registration, bio. Az összes többi útvonal védve van és bejelentkezve kell lenni, hogy kapjunk választ.

Ez az authentikáció a JsonWebToken segítségével van megoldva.

A backend abban az esetben ha szükséges token a kéréshez két választ tud küldeni.

* elutasítja mert unauthorized
* válaszol a kérésre